



Dive Into Android Networking: Adding Ethernet Connectivity

Benjamin Zores

ELCE 2012 – 6th November 2012 – Barcelona, Spain



Dive Into Android Networking: Adding Ethernet Connectivity About Me



ALCATEL LUCENT

ANDROID PLATFORM ARCHITECT

- Expert and Evangelist on Open Source Software.
- 9y experience on various multimedia/network embedded devices design.
- From low-level BSP integration to global applicative software architecture.

OPEN SOURCE

PROJECT FOUNDER, LEADER AND/OR CONTRIBUTOR FOR:

- [OpenBricks](#) Embedded Linux cross-build framework.
- [GeeXboX](#) Embedded multimedia HTPC distribution.
- [uShare](#) UPnP A/V and DLNA Media Server.
- [MPlayer](#) Linux media player application.

LINUX FOUNDATION CONFERENCES

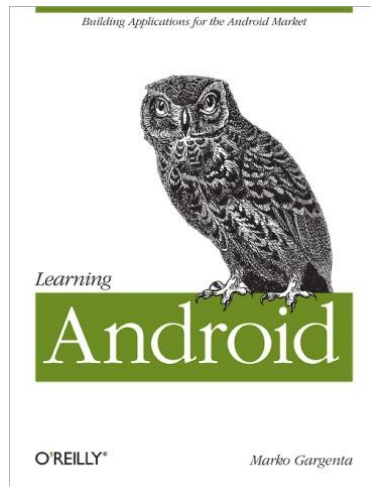
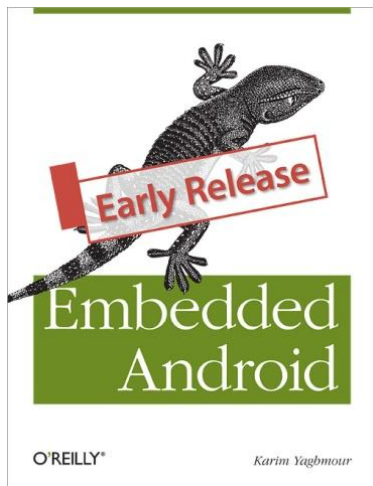
FORMER LINUX FOUNDATION'S EVENTS SPEAKER

- ELC 2010 [GeeXboX Enna: Embedded Media Center](#)
- ELC-E 2010 [State of Multimedia in 2010 Embedded Linux Devices](#)
- ELC-E 2011 [Linux Optimization Techniques: How Not to Be Slow ?](#)
- ABS 2012 [Android Device Porting Walkthrough](#)



Dive Into Android Networking: Adding Ethernet Connectivity

Bibliographical References



My Android bibles,
from my Android mentors:

Karim Yaghmour
Marko Gargenta

Followed by my own publications:
« **Discovering Android** »

Series of articles published in
GNU/Linux Magazine France



Dive Into Android Networking: Adding Ethernet Connectivity

Work Context



- Designing an **Enterprise Desktop IP Phone**.
- Differs heavily from usual Android devices:
 - Always connected, no battery
 - No Radio (GSM/CDMA).
 - No WiFi Station mode, AP only.
 - **LAN through Ethernet PHY/Switch.**
 - Always docked, no screen rotation.
 - No accelerometer, no GPS ...
 - => **Not a Smartphone**



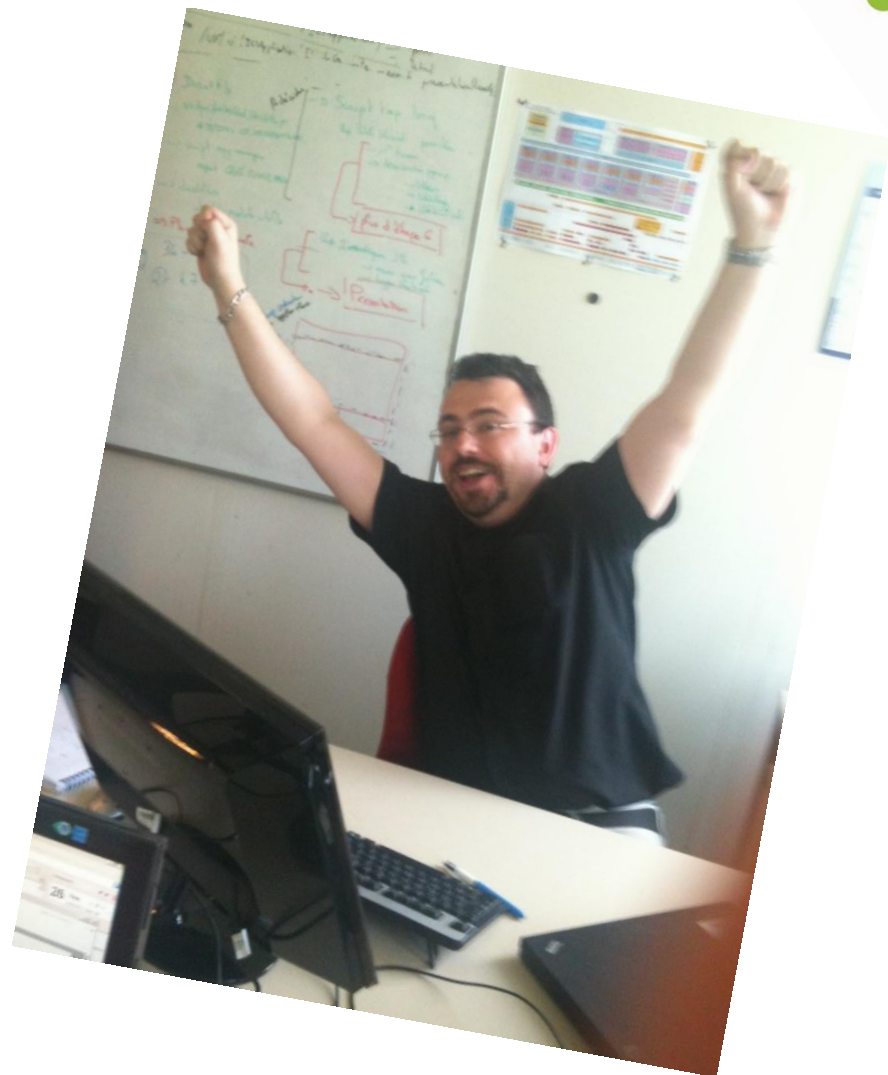
Dive Into Android Networking: Adding Ethernet Connectivity

Give Unto Caesar What is Caesar's ...



Most of the work
presented hereafter
is courtesy of

Fabien Brisset

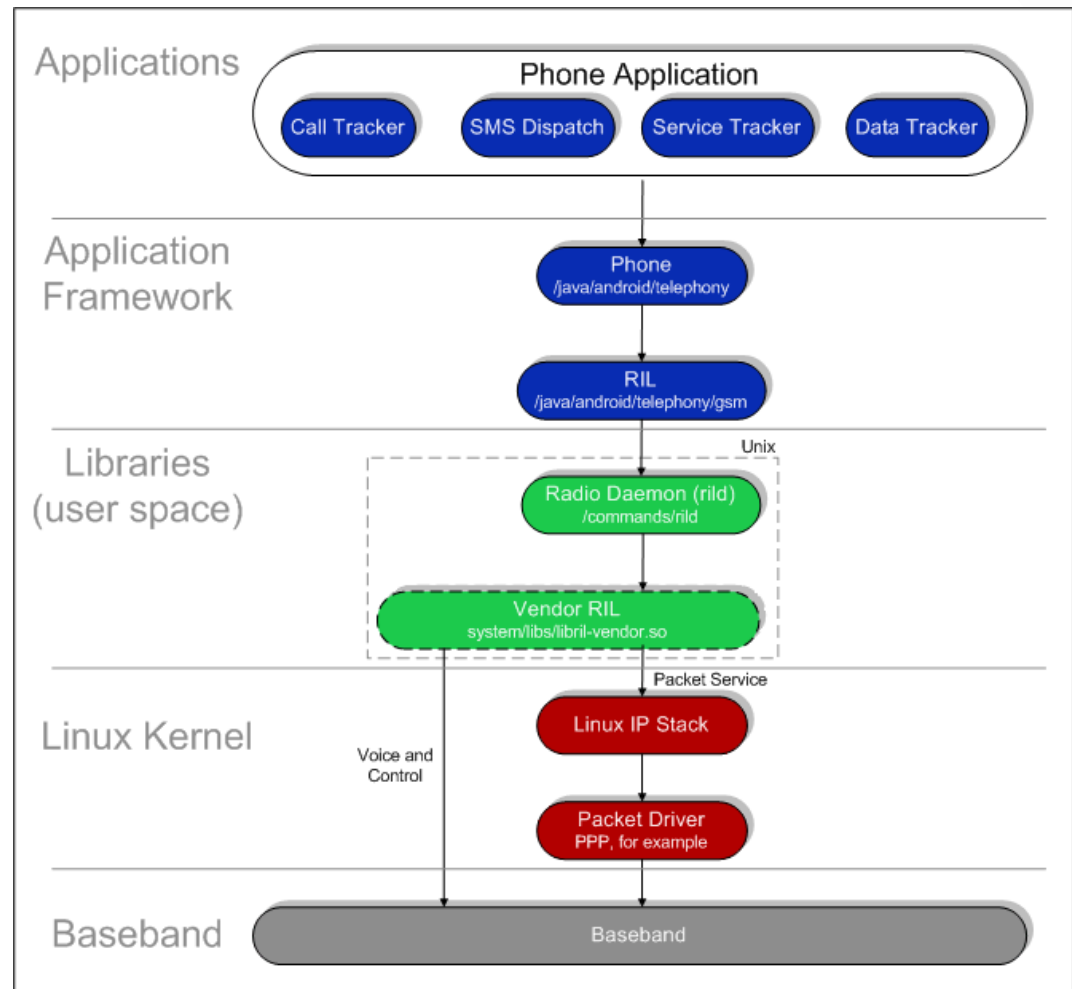


Dive Into Android Networking: Adding Ethernet Connectivity

Radio Layer Interface



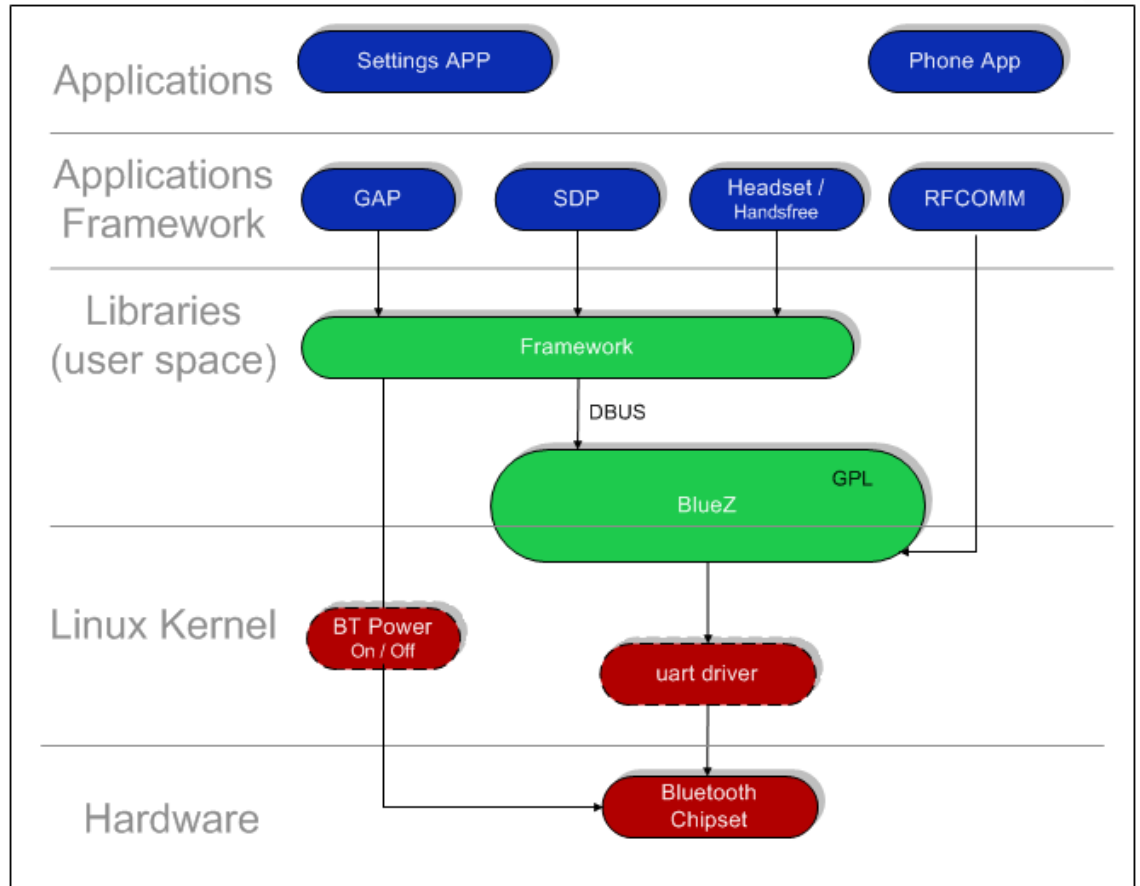
- Communicates with broadband processor for both voice and data.
- Uses **RIL** and proprietary **rild** to interface with system.
- Data connection is exported by **Connectivity Manager** through **TYPE_MOBILE_***



Dive Into Android Networking: Adding Ethernet Connectivity Bluetooth Interface

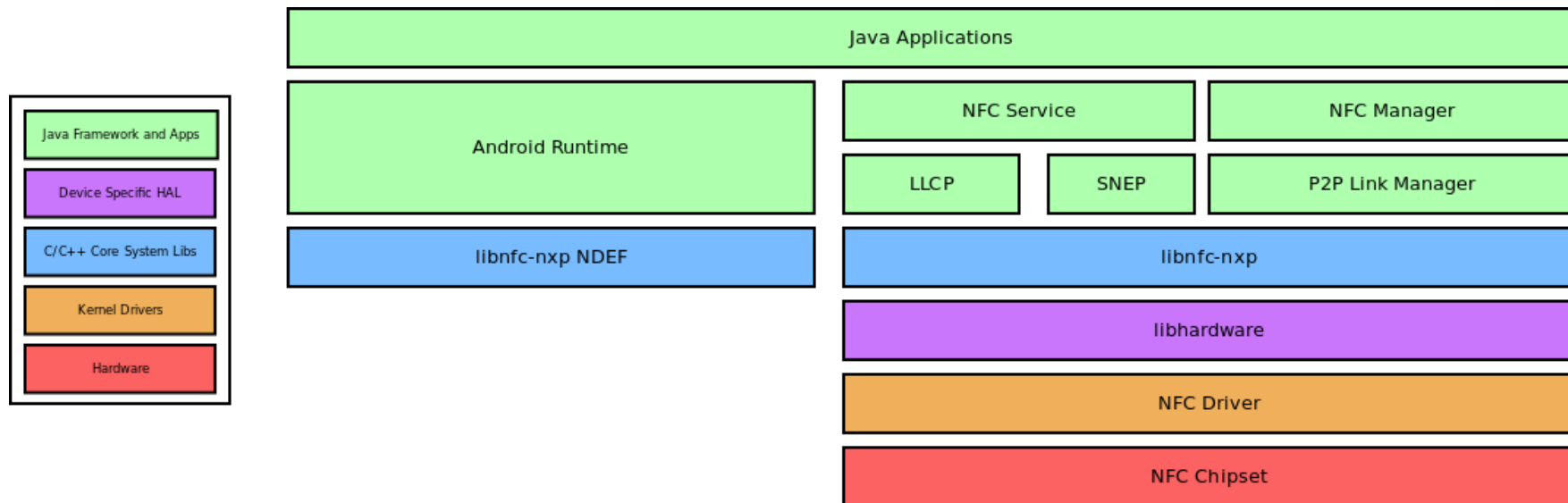


- Communicates with BT chipset through **BlueZ** and **libbluedroid**.
- Provides both audio and data management.
- Data connection is exported by **Connectivity Manager** through **TYPE_BLUETOOTH**.
- Interface with **netd** for tethering.



Dive Into Android Networking: Adding Ethernet Connectivity

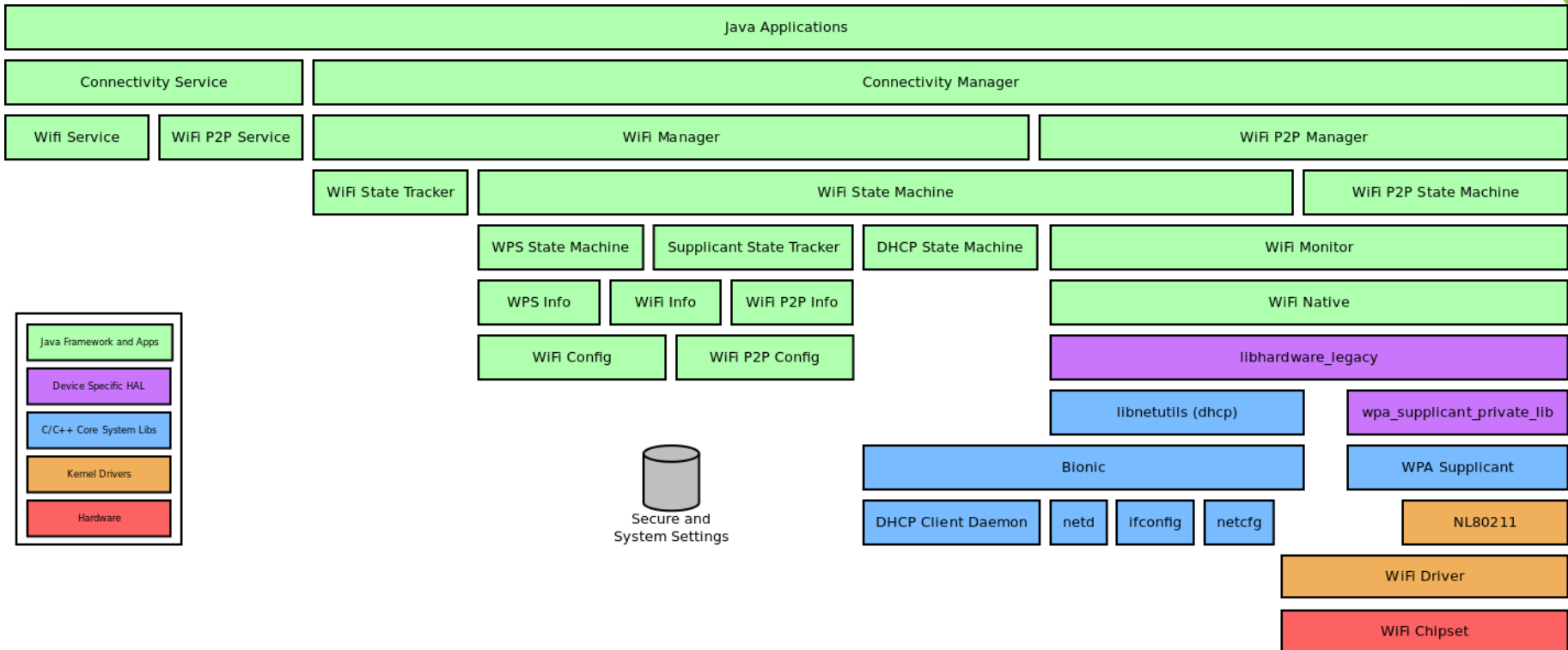
NFC Interface



- Introduced with **Ice Cream Sandwich** for **Near Field Communication**.
- Rely on **NFC HAL**.
- Currently only support chips from NXP (PN544).
- Uses **Android Beam** for P2P data exchange.
- Doesn't interface with **Connectivity Service/Manager**.

Dive Into Android Networking: Adding Ethernet Connectivity

WiFi Station/AP/P2P Interface



- Rely on **HAL** for specific driver interface with **JNI binding**.
- Data connection is exported by **Connectivity Manager** through **TYPE_WIFI**.
- WiFi configuration is stored in file system and SQLite database.

Dive Into Android Networking: Adding Ethernet Connectivity

Android Ethernet Upstream Status



- Ethernet is supported through native Linux kernel drivers.
- **ifconfig** / **netcfg** / **ping** commands work but remain at platform level.
- Regular **/etc/resolv.conf** DNS is not supported due to **Bionic** host resolution.
- Native system daemons (C/C++) support regular Linux networking API.
- Java framework services and apps rely on **Connectivity Manager** and have no clue what Ethernet route/connection actually means.
 - Except for some apps (e.g. **Browser**, which relies on native implementation).
- Barely no Android device features Ethernet
 - Except from some obscure Chinese tablets.
- Ethernet connection type exists in ICS API.
 - But with no **Connectivity Manager** or **Connectivity Service** implementation.

Dive Into Android Networking: Adding Ethernet Connectivity

Enterprise Requirements & ECM Status



- **Enterprise Requirements:**

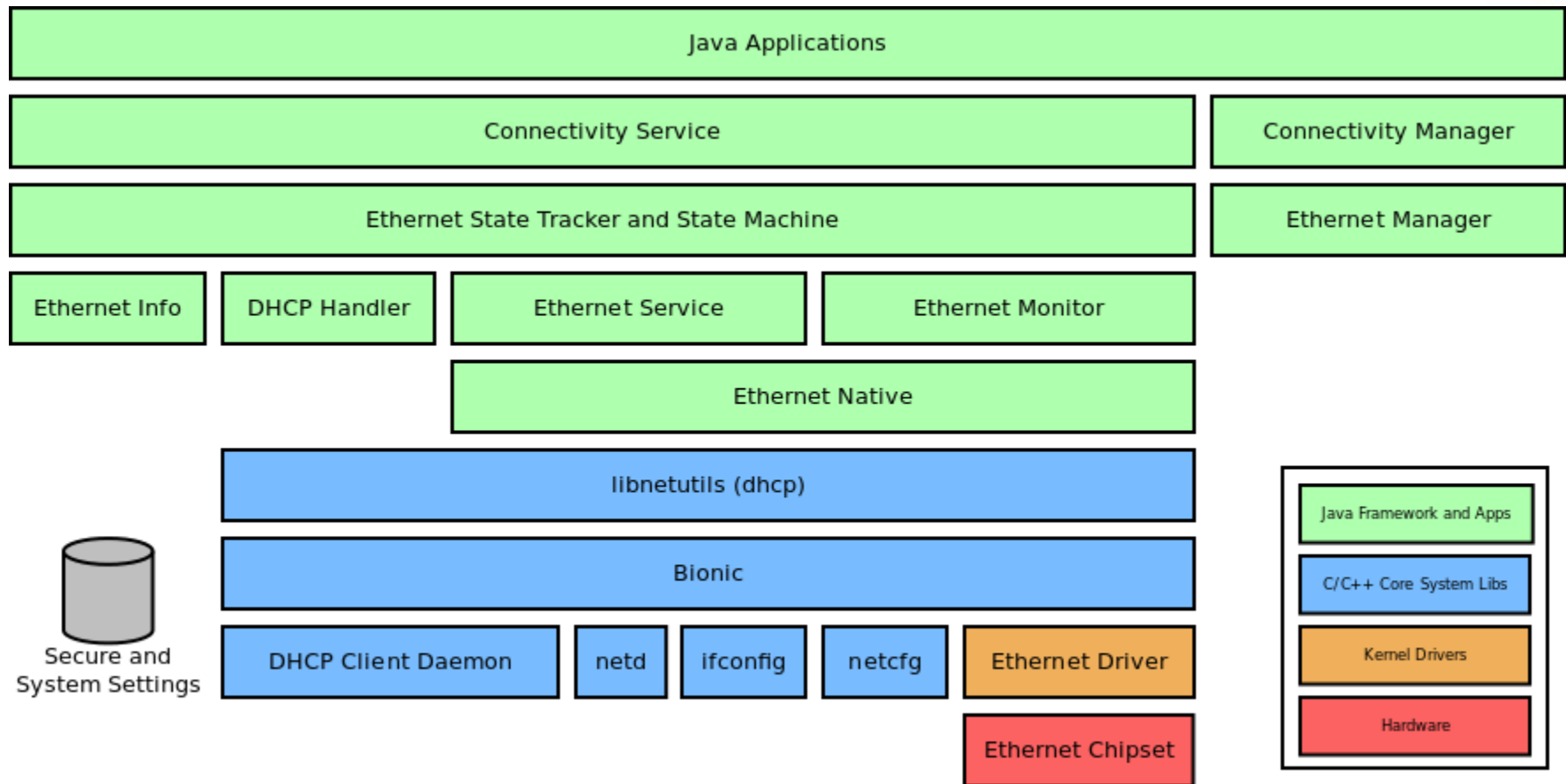
- Reliability: Ensuring data connection works in 100% cases for all possible applications.
- HTTP(S) seamless proxy support for all applications.
- Corporate firewalls prevents some services behavior (e.g. NTP).
 - Need to ensure everything stays behind the walls.
- Ethernet 802.1x authentication.

- **Ethernet Connectivity Manager (ECM) Status**

- ECM patch has been done by **Android-x86** team for netbooks.
 - **Not 100% accurate or sufficient.**

Dive Into Android Networking: Adding Ethernet Connectivity

Ethernet Interface – ECM Patch Status



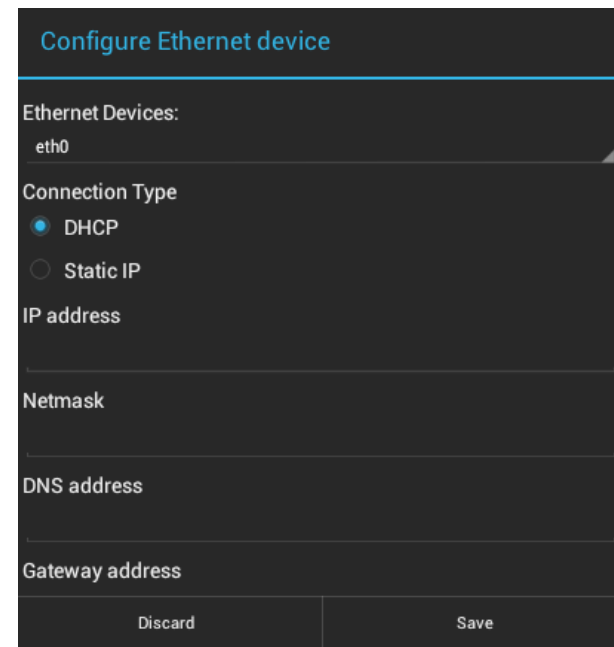
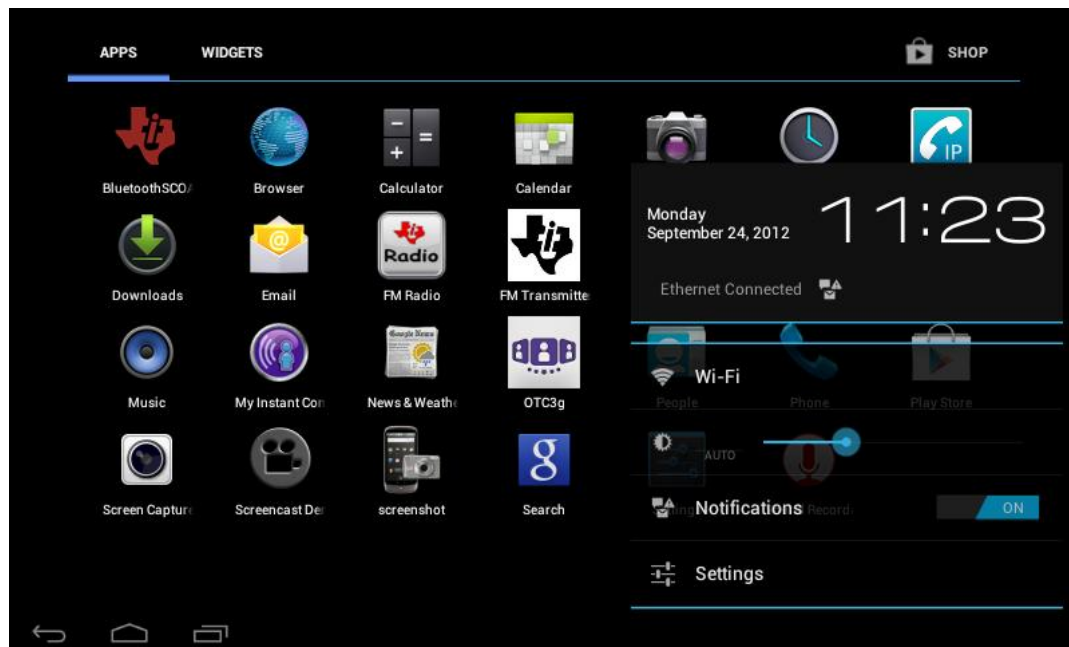
- Mismatch in implementation towards WiFi.
- Rely on **libnetutils** dhcp implementation instead of **DHCPd**.
- Not completely binded on **Connectivity Manager and Service**.

Dive Into Android Networking: Adding Ethernet Connectivity

Am I Connected or What ?



- Changes in Android framework, Settings app and System UI.
- Supports DHCP (ECM patch default) and Static IP (added) addressing.
- Connection status is available in notification bar.



Dive Into Android Networking: Adding Ethernet Connectivity ECM Patch Additions



- **Register Ethernet Service**

- In framework's `core/java/android/app/ContextImpl.java`:

```
registerService(ETHERNET_SERVICE, new ServiceFetcher() {  
    public Object createService(ContextImpl ctx) {  
        IBinder b = ServiceManager.getService(ETHERNET_SERVICE);  
        IEthernetManager srv = IEthernetManager.Stub.asInterface(b);  
        return new EthernetManager(srv, ctx.mMainThread.getHandler());  
    }  
});
```

- **Letting Connectivity Service know about Ethernet:**

- In framework's `services/java/com/android/server/ConnectivityService.java`:

```
[...] else if (networkType == ConnectivityManager.TYPE_ETHERNET)  
    usedNetworkType = ConnectivityManager.TYPE_ETHERNET;
```


Dive Into Android Networking: Adding Ethernet Connectivity ECM Patch Additions



- Forcing default network preferences:

- In framework's `core/java/android/net/ConnectivityManager.java`:

- `public static final int DEFAULT_NETWORK_PREFERENCE = TYPE_WIFI;`
 - + `public static final int DEFAULT_NETWORK_PREFERENCE = TYPE_ETHERNET;`

Issue: Android public API is being modified and `make update-api` is required.

- In framework's `packages/SettingsProvider/res/values/defaults.xml`:

- `<integer name="def_network_preference">1</integer>`
 - + `<integer name="def_network_preference">9</integer>`

- In framework's `services/java/com/android/server/EthernetService.java`:

- `Settings.Secure.putString(cr, Settings.Secure.ETHERNET_IFNAME, DevName[0]);`
 - + `Settings.Secure.putString(cr, Settings.Secure.ETHERNET_IFNAME, "eth0");`



**And now, bugs and workarounds
for various use cases:**

Making Ethernet 100% functional.

Dive Into Android Networking: Adding Ethernet Connectivity Connection Information



Status
Wi-Fi IP address Unavailable
Wi-Fi MAC address Unavailable
Ethernet IP address 172.25.52.225
Ethernet MAC address 00:04:9f:02:1a:10
Bluetooth address Unavailable
Serial number unknown
Up time 0:17:24

Symptom:

What is my Ethernet IP configuration or MAC address info ?

Dive Into Android Networking: Adding Ethernet Connectivity Connection Information – IP Address



- In Settings's `res/xml/device_info_status.xml`:

```
<Preference android:key="ethernet_ip_address"
    style="?android:attr/preferenceInformationStyle"
    android:title="@string/ethernet_advanced_ip_address_title"
    android:summary="@string/device_info_not_available"
    android:persistent="false" />
```

- In `src/com/android/settings/deviceinfo/Status.java`:

```
private void setEthernetIpAddressStatus() {

    EthernetManager mgr = getSystemService(ETHERNET_SERVICE);
    EthernetDevInfo info = mgr.getSavedConfig();

    Preference ip = findPreference("ethernet_ip_address");

    String addr = null;

    if (info != null) {
        if (info.getIpAddress() != null)
            addr = info.getIpAddress();
        else
            addr = SystemProperties.get("dhcp.eth0.ipaddress");
    }

    ip.setSummary(!TextUtils.isEmpty(addr) ? Addr :
        getString(R.string.status_unavailable));
}
```

Dive Into Android Networking: Adding Ethernet Connectivity Connection Information – MAC Address



- In framework's JNI code `core/jni/android_net_ethernet.cpp`:

```
{ "getInterfaceMacAddress", "()Ljava/lang/String;",  
  (void *) android_net_ethernet_getInterfaceMacAddress },  
  
[...]  
  
static jstring android_net_ethernet_getInterfaceMacAddress(JNIEnv *env, jobject clazz) {  
    struct ifreq ifr;  
    strcpy (ifr.ifr_name, "eth0");  
    strcpy (ifr.ifr_hwaddr.sa_data, "");  
  
    sock = socket (AF_INET, SOCK_STREAM, 0);  
  
    ioctl (sock, SIOCGIFHWADDR, &ifr);  
  
    ptr = (unsigned char *) ifr.ifr_hwaddr.sa_data;  
  
    snprintf (buf, 64, "%02x:%02x:%02x:%02x:%02x:%02x",  
              (ptr[0] & 0377), (ptr[1] & 0377), (ptr[2] & 0377),  
              (ptr[3] & 0377), (ptr[4] & 0377), (ptr[5] & 0377));  
  
    return env->NewStringUTF(buf);  
}
```

Dive Into Android Networking: Adding Ethernet Connectivity Connection Information – MAC Address



- In Settings' **res/xml/device_info_status.xml**:

```
<Preference android:key="ethernet_mac_address"
    style="?android:attr/preferenceInformationStyle"
    android:title="@string/status_ethernet_mac_address"
    android:summary="@string/device_info_not_available"
    android:persistent="false" />
```

- In Settings' **src/com/android/settings/deviceinfo/Status.java**:

```
private void setEthernetMacAddress() {

    EthernetManager mgr = getSystemService(ETHERNET_SERVICE);
    EthernetDevInfo info = mgr.getSavedConfig();

    Preference mac = findPreference("ethernet_mac_address");

    String addr = info == null ? null : info.getMacAddress();

    mac.setSummary(!TextUtils.isEmpty(addr) ? addr
        : getString(R.string.status_unavailable));
}
```


Dive Into Android Networking: Adding Ethernet Connectivity

Android DNS Entry Management



Symptom:

DHCP can't seem to provide me with valid DNS entries.

- Match Android process (AID) authorizations to update system properties in init's `init/property_service.c`:

```
{ "rw.",          AID_SYSTEM,    0 },
{ "net.",         AID_DHCP,      0 },
```

- In DHCPD's `dhcpcd-hooks/20-dns.conf`:

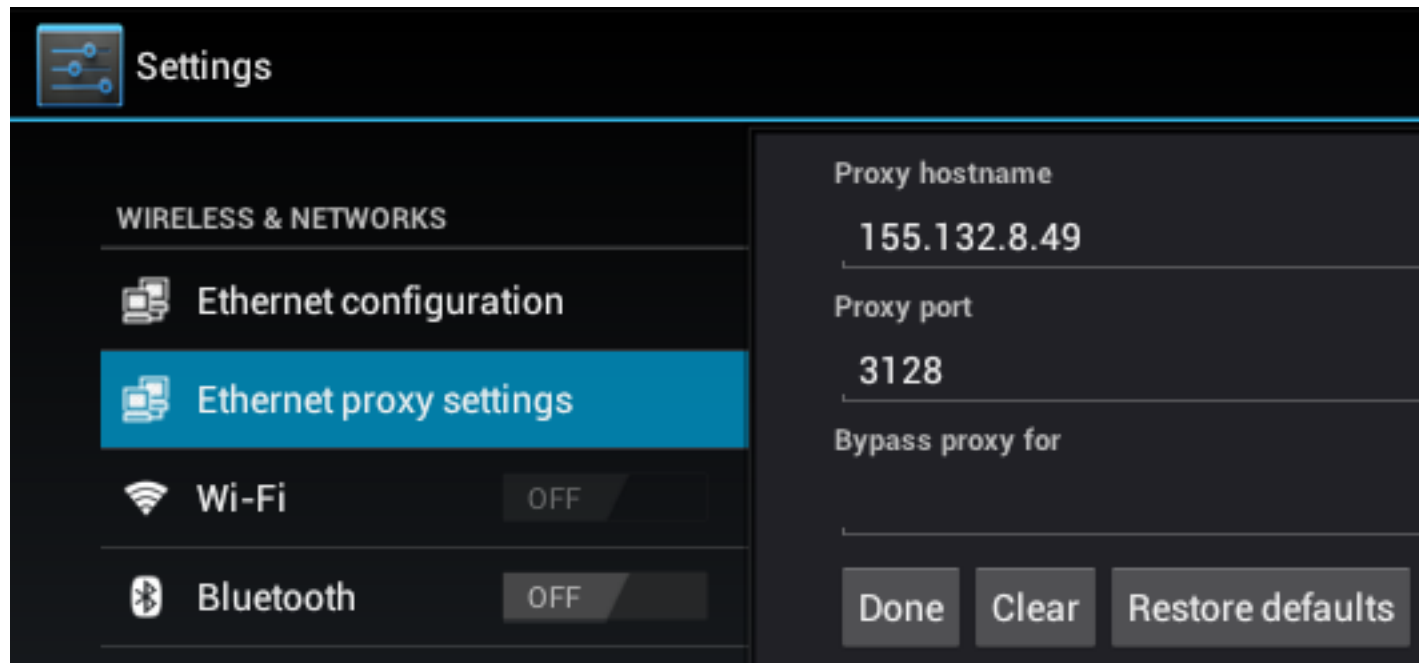
```
for dnsaddr in ${new_domain_name_servers}; do
    setprop dhcp.${interface}.dns${count} ${dnsaddr}
+   setprop net.dns${count} ${dnsaddr}
+   setprop net.${interface}.dns${count} ${dnsaddr}
    count=$((count + 1))
done
```

- In framework's `ethernet/java/android/net/ethernet/EthernetStateTracker.java`:

```
SystemProperties.set("net.dns1", mDhcpInfo.dns1);
SystemProperties.set("net." + mInterfaceName + ".dns1", mDhcpInfo.dns1);
```

Dive Into Android Networking: Adding Ethernet Connectivity

HTTP(S) Proxy



Symptom:

**I'm behind HTTP(S) proxy.
I need my apps to seamlessly know about that !**

Dive Into Android Networking: Adding Ethernet Connectivity HTTP(S) Proxy



- Overlay **frameworks/base/core/res/res/values/config.xml**:

```
<string name="config_default_proxy_host" translatable="false">a.b.c.d</string>
<integer name="config_default_proxy_port" translatable="false">8080</integer>
```

- In framework's **services/java/com/android/server/ConnectivityService.java**:

```
String proxyHost = context.getResources().getString(
    com.android.internal.R.string.config_default_proxy_host);

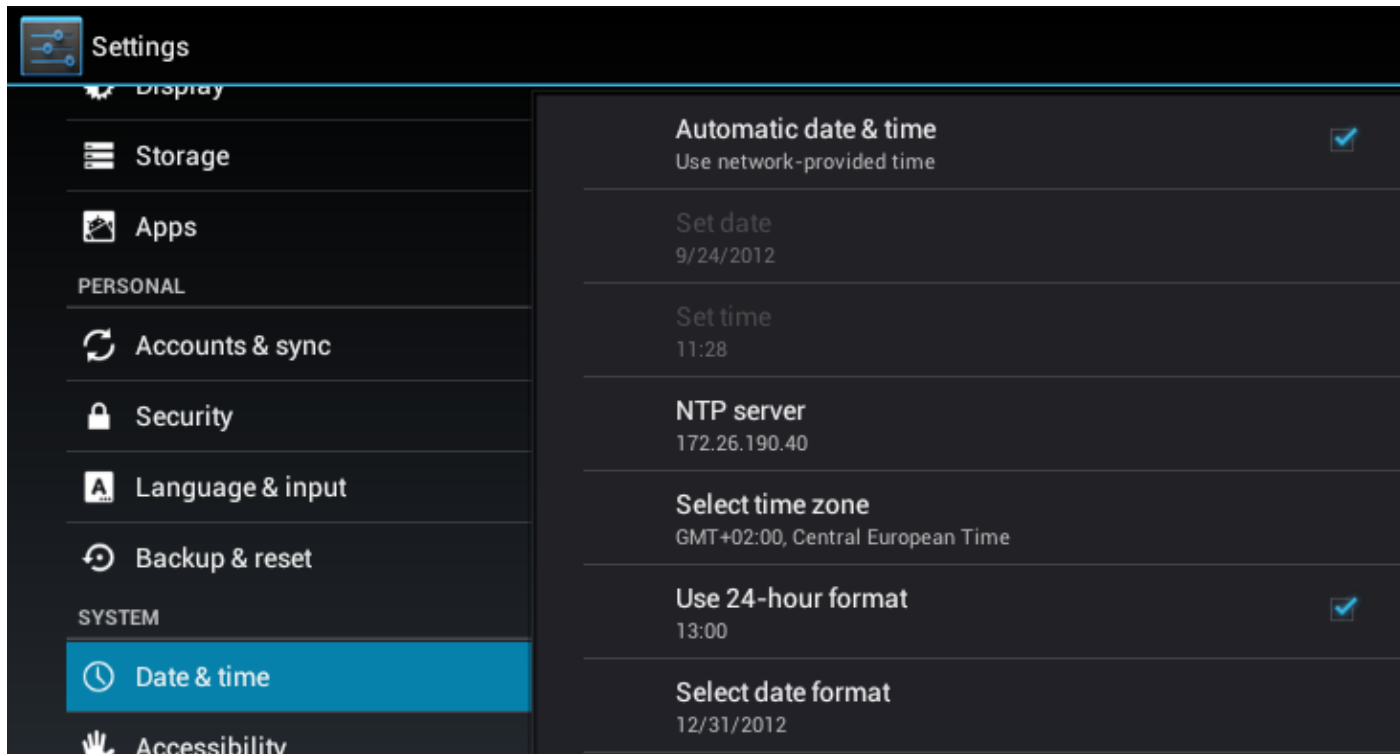
int proxyPort = context.getResources().getInteger(
    com.android.internal.R.integer.config_default_proxy_port);

mGlobalProxy = new ProxyProperties(proxyHost, proxyPort, null);

SystemProperties.set("net.http.proxy", proxyHost + ":" + proxyPort);
```

Dive Into Android Networking: Adding Ethernet Connectivity

Custom NTP Server



Symptom:

**I'm behind corporate firewall and can't do NTP request.
My company provides its internal NTP server.**

Dive Into Android Networking: Adding Ethernet Connectivity

Custom NTP Server



- Overlay **frameworks/base/core/res/res/values/config.xml**:

```
<!-- Remote server that can provide NTP responses. -->
<string translatable="false" name="config_ntpServer">a.b.c.d</string>

<!-- Timeout to wait for NTP server response. -->
<integer name="config_ntpTimeout">20000</integer>
```

- In framework's **core/java/android/util/NtpTrustedTime.java**:

```
-    final String defaultServer =
        res.getString(com.android.internal.R.string.config_ntpServer);

+    String defaultServer = Settings.System.getString(resolver, Settings.System.NTP_SERVER);
```

- In framework's **services/java/com/android/server/NetworkTimeUpdateService.java**:

- Force NTP update on Ethernet state change:

```
if (netInfo.getState() == NetworkInfo.State.CONNECTED &&
    netInfo.getType() == ConnectivityManager.TYPE_ETHERNET)
    mHandler.obtainMessage(EVENT_ETHERNET_CONNECTED).sendToTarget();

[...]

case EVENT_WIFI_CONNECTED:
case EVENT_ETHERNET_CONNECTED:
    onPollNetworkTime(msg.what);
```

Dive Into Android Networking: Adding Ethernet Connectivity Custom NTP Server



- In Settings's **res/xml/date_time_prefs.xml**:

```
<EditTextPreference
    android:title="@string/ntp_server_time"
    android:key="ntp_server"
    android:singleLine="true"
    android:summary="192.168.1.1"
    android:inputType="textUri"/>
```

- In Settings' **src/com/android/settings/DateTimeSettings.java**:

```
EditTextPreference pref = findPreference("ntp_server");

String server =
    Settings.System.getString(getContentResolver(), Settings.System.NTP_SERVER);

pref.setText(server);
pref.setSummary(server);
```


Dive Into Android Networking: Adding Ethernet Connectivity

Email



Symptom:

I can't download email attachments.

- **In `src/com/android/email/AttachmentInfo.java`:**

```
- if (networkType != ConnectivityManager.TYPE_WIFI) {  
+ if ((networkType != ConnectivityManager.TYPE_WIFI) &&  
    (networkType != ConnectivityManager.TYPE_ETHERNET)) {
```

- **In `src/com/android/email/service/AttachmentDownloadService.java`:**

```
- if (ecm.getActiveNetworkType() != ConnectivityManager.TYPE_WIFI) {  
+ if ((ecm.getActiveNetworkType() != ConnectivityManager.TYPE_WIFI) &&  
    (ecm.getActiveNetworkType() != ConnectivityManager.TYPE_ETHERNET)) {
```

Dive Into Android Networking: Adding Ethernet Connectivity

Connectivity Route



Symptom:

**I can't access to Google Play Store.
The application just crash !**

Dive Into Android Networking: Adding Ethernet Connectivity Connectivity Route



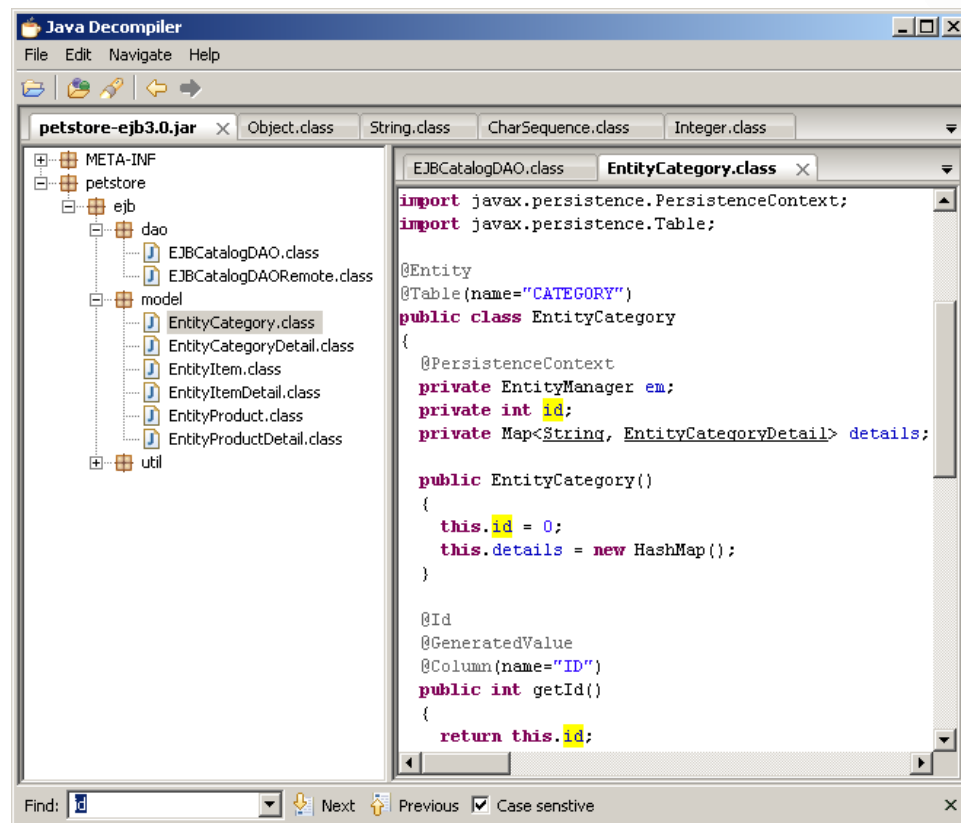
Reverse engineering apps may help:

- **Dex2jar**

- Extracts usable classes.jar from APK archives.
- See <http://code.google.com/p/dex2jar/>

- **Java Decompiler (jd-gui)**

- See <http://java.decompiler.free.fr/?q=jdgui>



Dive Into Android Networking: Adding Ethernet Connectivity Route



- **Many (certified?) apps assume that WiFi or mobile connection is always present.**
 - We need to trick the system for Ethernet.
 - But this is truly an ugly hack.
- **In framework's `services/java/com/android/server/ConnectivityService.java`:**

```
-      return getNetworkInfo(networkType, uid);
+      switch (networkType) {
+          case ConnectivityManager.TYPE_MOBILE:
+          case ConnectivityManager.TYPE_WIFI:
+          case ConnectivityManager.TYPE_WIMAX:
+              networkType = ConnectivityManager.TYPE_ETHERNET;
+              break;
+          default:
+              break;
+      }
+      return getNetworkInfo(networkType, uid);
+  }
```

- **Android shouldn't expose WiFi Manager API !**
 - Apps should always go through Connectivity Manager for all network information.
- **Limitations:**
 - Connectivity Manager API can't configure default connection type (currently hardcoded).
 - Too few information on existing devices' connectivity states.

Dive Into Android Networking: Adding Ethernet Connectivity Download Provider



Symptom:

My browser works fine but I just can't download files.

- In **DownloadProvider's** src/com/android/providers/downloads/DownloadInfo.java:
 - Add Ethernet connectivity support

```
case ConnectivityManager.TYPE_ETHERNET:
    return DownloadManager.Request.NETWORK_ETHERNET;

[...]

if (networkType == ConnectivityManager.TYPE_ETHERNET) {
    return NETWORK_OK; // anything goes over ethernet
}
```

Dive Into Android Networking: Adding Ethernet Connectivity Multimedia Streaming



Symptom:

My browser works fine but I can't play multimedia contents.

- **Force Chrome HTTP stack instead of Android HTTP stack.**
 - By default, **Stagefright** can't bypass proxy (see [frameworks/base/media/libstagefright/Android.mk](#))
 - Overlay in your device's **device.mk** file: `HTTP := chrome`

- **In framework's [media/libstagefright/chromium_http/support.cpp](#):**

```
char value[PROPERTY_VALUE_MAX];
property_get("net.http.proxy", value, "Unknown");

net::ProxyConfigServiceAndroid cfg = new net::ProxyConfigServiceAndroid();
if (strcmp(value_proxy, "Unknown") != 0) {
    std::string proxy = value;
    cfg->UpdateProxySettings(proxy, "");
}

set_proxy_service(net::ProxyService::CreateWithoutProxyResolver(cfg, net_log()));
```

- **In framework's [services/java/com/android/server/ConnectivityService.java](#):**

```
SystemProperties.set("net.proxy", host + ":" + port);
SystemProperties.set("rw.HTTP_PROXY", "http://" + host + ":" + port);
```


Dive Into Android Networking: Adding Ethernet Connectivity Phone / SIP VoIP



Symptom:

I can do VoIP SIP calls over WiFi but not over Ethernet.

- Overlay **frameworks/base/core/res/res/values/config.xml**:

```
<bool name="config_sip_ethernet">true</bool>
```

- In framework's **voip/java/android/net/sip/SipManager.java**:

```
public static boolean isSipEthernet(Context context) {  
    return context.getResources().getBoolean(com.android.internal.R.bool.config_sip_ethernet);  
}
```

- In framework's **voip/java/com/android/server/sip/SipService.java**:

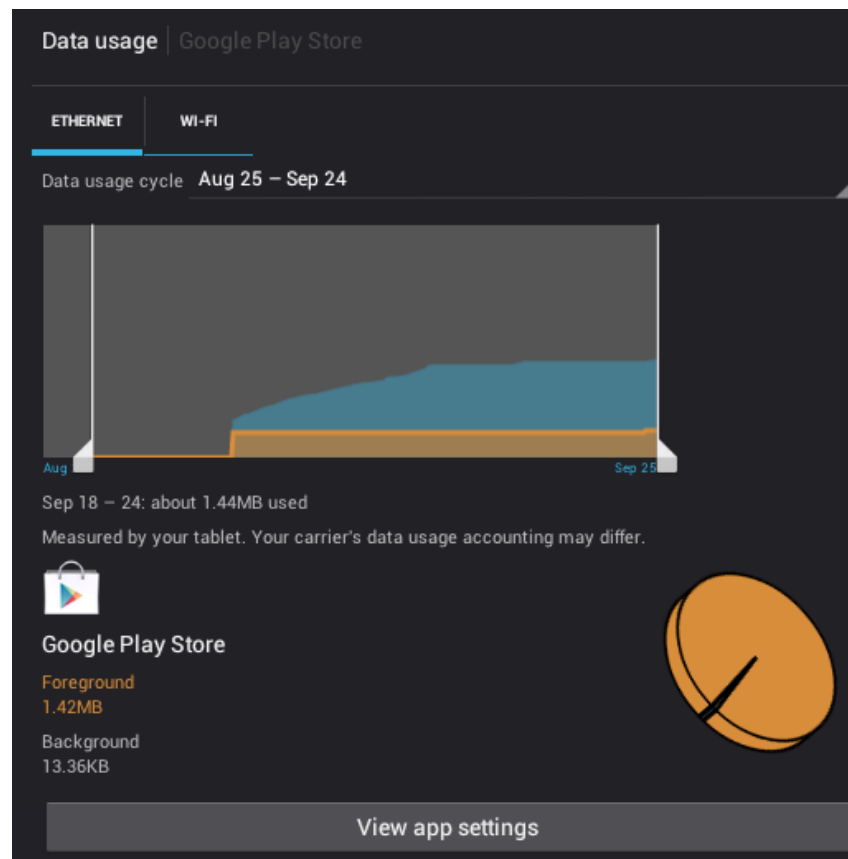
```
boolean mSipOnWifiOnly = SipManager.isSipWifiOnly(context);  
boolean mSipEthernet = SipManager.isSipEthernet(context);  
  
boolean connected = (info != null && info.isConnected()) &&  
    ((!mSipOnWifiOnly || info.getType() == ConnectivityManager.TYPE_WIFI) ||  
    ( mSipEthernet    && info.getType() == ConnectivityManager.TYPE_ETHERNET));
```

- In Phone's **src/com/android/phone/SipCallOptionHandler.java**:

```
return ((ni.getType() == ConnectivityManager.TYPE_WIFI) || !SipManager.isSipWifiOnly(this)) ||  
    (ni.getType() == ConnectivityManager.TYPE_ETHERNET) && SipManager.isSipEthernet(this));
```

Dive Into Android Networking: Adding Ethernet Connectivity

Network Statistics



Symptom:

How much data did I use ?
Where are my network statistics ?

Dive Into Android Networking: Adding Ethernet Connectivity Network Statistics



- Overlay **frameworks/base/core/res/res/values/config.xml**:

```
<!-- Set of NetworkInfo.getType() that reflect data usage. -->
<integer-array translatable="false" name="config_data_usage_network_types">
    <item>9</item> <!-- TYPE_MOBILE_ETHERNET -->
</integer-array>

<!-- The default iface on which to monitor data use -->
<string name="config_datause_iface">eth0</string>
```

- Update **Logtags** samples in framework's **services/java/com/android/server/EventLogTags.logtags**:

```
51102 netstats_ethernet_sample
(dev_rx_bytes|2|2), (dev_tx_bytes|2|2), (dev_rx_pkts|2|1), (dev_tx_pkts|2|1), (xt_rx_bytes|2|2),
(xt_tx_bytes|2|2), (xt_rx_pkts|2|1), (xt_tx_pkts|2|1), (uid_rx_bytes|2|2), (uid_tx_bytes|2|2), (u
id_rx_pkts|2|1), (uid_tx_pkts|2|1), (trusted_time|2|3), (dev_history_start|2|3)
```

- In framework's **ethernet/java/android/net/ethernet/EthernetStateTracker.java**:

- One need to add support for **LinkProperties**

```
LinkProperties mLinkProperties = mDhcpInfo.makeLinkProperties();
mLinkProperties.setInterfaceName("eth0");
```

Dive Into Android Networking: Adding Ethernet Connectivity Network Statistics



- In framework's `services/java/com/android/server/net/NetworkStatsService.java`:

- Need to collect Ethernet samples.

```
import static android.net.NetworkTemplate.buildTemplateEthernet;

NetworkTemplate template = buildTemplateEthernet();

devTotal = getSummaryForNetworkDev(template, start, end).getTotal(devTotal);
devHistoryStart = getHistoryStartLocked(template, mNetworkDevStats);
xtTotal = getSummaryForNetworkXt(template, start, end).getTotal(xtTotal);
uidTotal = getSummaryForAllUid(template, start, end, false).getTotal(uidTotal);

EventLogTags.writeNetstatsEthernetSample(
    devTotal.rxBytes, devTotal.rxPackets, devTotal.txBytes, devTotal.txPackets,
    xtTotal.rxBytes, xtTotal.rxPackets, xtTotal.txBytes, xtTotal.txPackets,
    uidTotal.rxBytes, uidTotal.rxPackets, uidTotal.txBytes, uidTotal.txPackets,
    trustedTime, devHistoryStart);
```

- Adding `ConnectivityManager.TYPE_ETHERNET` support to Monkey's `NetworkMonitor`

- Used to display time spent proceeding data from Ethernet interface.

Dive Into Android Networking: Adding Ethernet Connectivity

Next Steps ?



- Properly redesign the **ECM patch** to match WiFi architecture.
- Port from **Ice Cream Sandwich** to **Jelly Bean**
 - Check if original ECM patch has been updated since early 2012.
 - Current changeset with extra features is 504 kB big.
- Design Ethernet HAL for **802.1x / WPA support**.
- Contribute / upstream to **Linaro** ?
- And then to **Google** ?

Dive Into Android Networking: Adding Ethernet Connectivity

Thanks



Thank You



AT
THE
SPEED
OF
IDEAS™

www.alcatel-lucent.com